Structured Concept Discovery: Theory and Methods

Darrell Conklin
Department of Computing and Information Science
Queen's University
Kingston, Ontario, Canada K7L 3N6
conklin@qucis.queensu.ca

Technical Report No. 94-366

June 1994

Abstract

The field of knowledge discovery is concerned with the theory and processes involved in finding and representing patterns and regularities previously unknown. A new generation of knowledge discovery tools now deals with *structured concepts*: these capture associations between relations among the components of structured objects. This paper outlines a logic used to express structured concepts, and surveys a number of systems performing structured concept discovery. The paper concludes with a discussion of important future research directions for the field.

Contents

1 Introduction and motivations

The field of knowledge discovery is concerned with the theory and processes involved in finding and representing things previously unknown. Empirical discovery systems search for regularities and detect interesting and useful patterns from incoming data. There is increasing interest in this field, due to the growing number of giga- or tera-byte databases containing largely factual information. It is becoming clear that techniques are needed for structuring, managing, compressing, assimilating and extracting general knowledge from these data.

As an abstract motivation for knowledge discovery, imagine that we have created a robot and set it free in a new environment. To survive, the robot must have the ability to recognize new things, particularly those hostile to its survival. While we, as system designers, will have provided the robot with a language for representing facts and knowledge, we cannot have anticipated all types of situations the robot will find itself in. One tactic the robot might use is to simply remember factual descriptions of all objects encountered, matching new objects to those in the database and making predictions on the basis of similarity. This process, known as *reasoning by analogy*, will become increasingly tedious as the size of the database grows. If the robot can discover associations and regularities in the data, these can be used both to make expedient predictions about new objects and to index database objects for efficient retrieval.

Current knowledge discovery systems (Matheus et al., 1993) use a highly restricted knowledge representation language for representing patterns and regularities: they can only capture associations among different features of objects. A new generation of knowledge discovery tools now deals with *structured concepts*: they capture associations between relations among the components of objects. Such tools are essential, for example, in data analysis and object recognition tasks where the features of an object's components alone are insufficient to determine its classification or function. A description of the interrelationships among parts is crucial.

This paper has two purposes. One is to present a logic in which all structured concepts discovery systems can be discussed and related. Another purpose of this paper is to survey a number of operational systems which perform discovery of structured concepts. Although Thompson and Langley (1991) provided a survey of several systems, with recent advancements there is a need for a broader discussion which includes theoretical foundations, and surveys of newer systems, miscellaneous systems, and application domains. This paper concludes with a discussion of critical future research issues in the field.

2 Structured concepts: theoretical foundations

A *concept*, defined extensionally, is simply a set of things in a universe of discourse. Machine learning systems construct and manipulate intentional concept descriptions. Knowledge representation is crucial for a machine learning system, as the set of possible learnable concepts is restricted to the set of concepts the system can construct or represent. Much of the early research on machine learning was restricted to a knowledge representation formally equivalent to propositional logic. Objects are described by sets of *features*, or attribute/value pairs. Recent work has seen a shift to more powerful first-order formalisms capable of describing structured objects and relational concepts.

An early paper on structured concept learning (Dietterich and Michalski, 1981) lamented the fact that each research group was using its own notation and terminology, making it hard to relate different systems performing similar learning tasks. The situation for current structured concept discovery systems is not much improved. This is especially regrettable in the light of theoretical advancements in the parallel field of knowledge representation. This section will outline a unified notation and theory to which all current structured concept discovery systems can be related.

2.1 Logical concepts

A *first-order language* comprises variable, constant, and predicate symbols with their arities.[1] A *term* in a given language is either a variable or a constant. An *atom* is an expression $p(t_1, \ldots, t_n)$, where $t_1, \ldots, t_n$ are terms, and p is an n–ary predicate symbol. An atom is *ground* if it does not contain any variables. A *literal* is an atom (A) or the negation of an atom (`not` A). A *well-formed-formula* in the language is formed using literals and operators such as conjunction, negation, and implication, and the quantifiers $\forall$ and $\exists$. A *sentence* is a closed well-formed-formula (all variables quantified). A *concept* can be represented by a unary λ predicate[2]

$$\lambda x.E, \tag{1}$$

where x is the only free variable in the well-formed-formula E.

The extension of a concept is all things in a universe of discourse such that when their denoting constant is substituted for the variable x in (1), E is satisfied. More formally, an *interpretation* I of a first-order language is given by a structure

$$\langle \mathcal{D}, [\![\cdot]\!]^I_u \rangle, \tag{2}$$

where $\mathcal{D}$ is a universe of discourse, and $[\![\cdot]\!]^I_u$ is a function which maps n-ary predicates to elements of $\mathcal{D}^n \rightarrow \{true, false\}$, sentences to $\{true, false\}$, and variables to elements of $\mathcal{D}$ via the variable assignment function u (see Table 1).

A *theory* is a set of sentences in a first-order language. A *model* for a theory is an interpretation which maps every sentence in the set to *true*. A concept C is *instantiated* by an object a, with respect to a theory $\mathcal{T}$, if

$$\mathcal{T} \models C(a), \tag{3}$$

that is, if $C(a)$ is *true* in all models of $\mathcal{T}$. A concept C is said to *subsume* a concept D, with respect to a theory $\mathcal{T}$, written $C \succeq_{\mathcal{T}} D$, if C is instantiated whenever D is:

$$\mathcal{T} \models \forall \mathtt{x}\, D(\mathtt{x}) \Rightarrow C(\mathtt{x}). \tag{4}$$

[1]See (Lloyd, 1987; Genesereth and Nilsson, 1988) for detailed discussions of first-order logic. Here function-free first-order logic is described.

[2]This is the standard definition of a concept used by the knowledge representation community (Woods, 1991; Woods and Schmolze, 1992) and the machine discovery community (Zytkow, 1993). See also (Hayes, 1985) for related knowledge representation issues. See (Tennent, 1991) for a review of the λ calculus.

Two concepts C and D are (semantically) *equivalent* with respect to a theory $\mathcal{T}$, written $C \equiv_{\mathcal{T}} D$, if they co-subsume each other:

$$\mathcal{T} \models \forall \mathtt{x}\, D(\mathtt{x}) \Leftrightarrow C(\mathtt{x}). \tag{5}$$

Two concepts C and D are (semantically) *disjoint* with respect to a theory $\mathcal{T}$, if instantiation of one implies noninstantiation of the other.

Before we continue, a word about the use of concepts as a hypothesis space for knowledge discovery is in order. Concepts are good tools for representing associations between symbolic attributes of objects. As pointed out by Zytkow (1993), they have limitations and are not the best formalism for capturing feature associations or numerical regularities. For example, the statement

$$\forall x\, \mathtt{height}(x) = c \times \mathtt{weight}(x)$$

represents a regularity between two attributes `height` and `weight` which holds for all objects. This regularity, however, is poorly represented by concepts, however, as a new concept must be produced for each value in the range of the `height` function.

2.1.1 A simple example

As a simple illustration of the theory outlined in this section, consider the following example, adapted from (Buntine, 1988). Consider the simple theory $\mathcal{T}$ comprising the two sentences

$$\forall \mathtt{x}\, \mathtt{cat(x)} \Rightarrow \mathtt{pet(x)}$$
$$\forall \mathtt{x}\, \mathtt{dog(x)} \Rightarrow \mathtt{pet(x)}.$$

In any model of $\mathcal{T}$, cats and dogs are pets. Let the concept C1 be

$$\lambda \mathtt{x.small(x)} \wedge \mathtt{fluffy(x)} \wedge \mathtt{dog(x)}.$$

The meaning of this concept (see Table 1; rules 9, 8, and 1) is a function that returns *true* for only those objects that are fluffy, small dogs. Let the concept C2 be

$$\lambda \mathtt{x.fluffy(x)} \wedge \mathtt{cat(x)},$$

that is, the concept of "fluffy cats". Then the concept of "fluffy pets"

$$\lambda \mathtt{x.fluffy(x)} \wedge \mathtt{pet(x)}$$

subsumes both C1 and C2 with respect to $\mathcal{T}$. It subsumes C1 because all small, fluffy dogs are also small fluffy pets, and hence they are also fluffy pets. It subsumes C2 because all fluffy cats are also fluffy pets. These subsumption relations can be verified by substituting for C and D in Expression 4.

Table 1: The formal semantics of a restricted first-order logic.

1. $[\![p(v_1,\ldots,v_n)]\!]^I_u = [\![p]\!]^I_u([\![v_1]\!]^I_u,\ldots,[\![v_n]\!]^I_u)$ *(predicate application)*
2. $[\![\texttt{true}]\!]^I_u = true$ *(truth constant)*
3. $[\![\texttt{false}]\!]^I_u = false$ *(falsity constant)*
4. $[\![\texttt{not}\ P]\!]^I_u = true$ iff $[\![P]\!]^I_u = false$ *(negation)*
5. $[\![\forall v\ P]\!]^I_u = true$ iff $[\![P]\!]^I_{(u|v\mapsto c)} = true$ for all $c \in \mathcal{D}$ *(universal quantification)*
6. $[\![\exists v\ P]\!]^I_u = true$ iff $[\![P]\!]^I_{(u|v\mapsto c)} = true$ for some $c \in \mathcal{D}$ *(existential quantification)*
7. $[\![\exists^* \overline{v}^m\ P]\!]^I_u = true$ iff $[\![P]\!]^I_{u'} = true$, where
   $u' = (u|v_1 \mapsto c_1,\ldots,v_m \mapsto c_m)$, $\{c_1,\ldots,c_m\} \subseteq \mathcal{D}$, and
   $c_1 \neq c_2 \neq \ldots \neq c_m$ *(restricted existential quantification)*
8. $[\![P \wedge Q]\!]^I_u = true$ if $[\![P]\!]^I_u = true$ and $[\![Q]\!]^I_u = true$ *(conjunction)*
9. $[\![\lambda x.E]\!]^I_u = f \in \mathcal{D} \rightarrow \{true, false\}$, where
   $f(a) = [\![E]\!]^I_{(u|x\mapsto a)}$ *(concept description)*
10. $[\![P \Rightarrow Q]\!]^I_u = true$ if $[\![P]\!]^I_u = false$ or $[\![Q]\!]^I_u = true$ *(implication)*

2.2 Structured concepts

Machine learning systems search over sets of possible concept definitions. This underlying first-order language is usually referred to as a *hypothesis space*. The hypothesis space used in the simple example above was weak, allowing only unary predicates. This section will develop and discuss a more powerful hypothesis space for *structured concepts*.

A *structured object* comprises parts along with defined relations among these parts.[3] A *structured concept* has structured objects in its extension. Haussler (1989) has given a precise characterization of a hypothesis space for structured concepts. This will be presented here, in a form slightly modified to allow for new theoretical and application developments.

An *existential conjunctive concept* is a closed sentence of the form

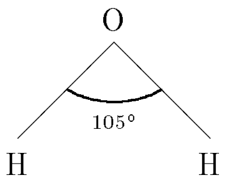
$$(\exists^* \overline{v}^m)\ p_1 \wedge \ldots \wedge p_k \tag{6}$$

where $\overline{v}^m$ denotes a set of m variables $v_1,\ldots,v_m$, each p_i is a literal with variables from $\overline{v}^m$ as arguments.[4] Expression (6) does not quite conform to the definition of a concept given by Expression (1). It can be satisfied by an interpretation, but is not a predicate and therefore does not have an extension. This is easily corrected by wrapping the expression in a λ form, introducing a variable $\mathtt{x}$ and a new distinguished predicate π:

$$\lambda\mathtt{x}.(\exists^* \overline{v}^m)\ \Pi(\mathtt{x}, \overline{v}^m) \wedge p_1 \wedge \ldots \wedge p_k \tag{7}$$

[3]Later on we will encounter *composite* structured objects, which recursively comprise other structured objects as parts.

[4]Haussler uses *functional relations*: the p_i are formulae of the form $p(\overline{v}^n) = y$ where p is an n–ary function. The effective hypothesis space is similar, although here quantitative attributes are not expressed. An n-ary functional relation p with $|ran(p)| = k$ will give rise to k n-ary predicates. A function with an infinite range requires special treatment.



```
λx.∃* p1 p2 p3
      π(x,p1) ∧
      π(x,p2) ∧
      π(x,p3) ∧
      hydrogen(p1) ∧
      hydrogen(p2) ∧
      oxygen(p3) ∧
      bonded(p1,p3) ∧
      bonded(p2,p3) ∧
      not bonded(p1,p2) ∧
      angular(p1,p2,p3).
```

Figure 1: A water molecule. Left: molecular graph depicting two hydrogens bonded to one oxygen atom. Right: a structured concept for the water molecule.

where $\Pi(\mathtt{x}, \overline{v}^m)$ is shorthand for the expression

$$\pi(\mathtt{x}, v_1) \wedge \ldots \wedge \pi(\mathtt{x}, v_m) \tag{8}$$

The predicate π, used to decompose a single object into parts, is called a *fundamental relation* (Thompson and Langley, 1989) and its intended meaning is usually the "immediate part of" relation. Thus the meaning of (7) is "all objects containing at least these parts, in these particular relationships".[5] The predicate symbol π can only appear in this restricted circumstance.

The notation $(\exists^* \overline{v}^m)$ in (6) (see Table 1) means that the variables $v_1, \ldots, v_m$ refer to *distinct* objects (see Table 1). Expression (7) captures the semantics of structured concepts as used by many of the systems surveyed in Section 3. Stepp (1987) has called this a *contains* versus an *is* semantics. In the latter, the concept is satisfied by objects that have *only* the described parts. An *is* semantics can be encoded in two ways. One is to conjoin to the concept expression a sentence asserting that the object has no other parts. Another idea, not suggested by Stepp, is to form an m-ary predicate from a concept with m parts. The translation between the two forms is mechanical: in this paper I assume a *contains* semantics throughout.

It may be useful at this point to visualize a structured concept — with only one binary relation in the first-order language — as a labelled graph: the parts being the vertices and relations between parts captured by edges. Figure 1 illustrates a structured concept for a water molecule. It has three parts — two hydrogen and one oxygen — decomposed by

[5] Haussler calls the objects "scenes", in deference to the pioneering work of Winston (1975).

the π relation. The quantitative angle measurement (105°) is expressed as a qualitative *angular* spatial relationship (as opposed to a *linear* relationship). At the left is a graph representation of the concept: to the right is its representation in first-order logic.[6] As we shall see, structured concept discovery systems can use various internal representations to capture the standard semantics given in this section: the restricted first-order logic used here provides a unified notation.

2.3 Subsumption of concepts

Computing whether one concept subsumes — is more general than — another (recall Expression 4) is a central facility of all structured concept discovery systems. A common method for subsumption computation is based on the so–called *θ-subsumption* (Plotkin, 1971; Gottlob, 1987). I assume familiarity with the idea of substitutions. θ-subsumption usually applies to logical *clauses* rather than concepts: the modifications made for the following definition are very slight.

Definition 1 *A structured concept $\lambda x.(\exists^* \overline{v}^m) C$* θ-subsumes *a structured concept $\lambda x.(\exists^* \overline{w}^n) D$ if there exists an injective substitution $\theta : \overline{v}^m \rightarrow \overline{w}^n$ such that $C\theta \subseteq D$.*

Proposition 1 *For any two structured concepts C and D, $C \succeq_{\emptyset} D$ if and only if C θ-subsumes D.*

(Note that the background theory is empty in $\succeq_{\emptyset}$). The proof of this completeness result is straightforward and related proofs can be found in (Haussler, 1989; Buntine, 1988). The substitution θ can be restricted to be injective due to the $\exists^*$ quantifier. The weakness of θ-subsumption, as pointed out by Buntine (1988), is that it assumes that the prior theory $\mathcal{T}$ is empty. Although there have been many successes in machine learning in theory–free applications, recently researchers have come to realize that theory and knowledge representation is indispensable in many domains (Stepp and Michalski, 1986).

From Expression (4), and the soundness and completeness of first-order logic, it follows that the subsumption relationship $C \succeq_{\mathcal{T}} D$ can be validated using theorem proving techniques. If the set of sentences

$$\mathcal{T} \cup \{\exists \mathtt{x}\; D(\mathtt{x}) \wedge \mathtt{not}\; C(\mathtt{x})\}$$

is unsatisfiable, then $C \succeq_{\mathcal{T}} D$. Unfortunately, in the presence of an arbitrary theory $\mathcal{T}$, this computation is only semi-decidable. It is, however, decidable in restricted first-order languages such as Datalog programs (sets of Horn clauses without function symbols). In such theories subsumption can be computed by *saturating* the concepts with background knowledge — expanding both concepts to semantically equivalent ones by applying all implications in the theory (Rouveirol, 1994) — and testing for standard θ-subsumption.

The instantiation problem — determining whether an object is an instance of a structured concept (recall Expression 3) — is very similar to the subsumption problem. In fact,

[6]Note the semantics assigned to the graph structure in Figure 1; the absence of an edge means that the vertices are *not* related by that relation (e.g., the two hydrogens are not bonded). For brevity, negated *angular* relationships are omitted from the figure.

many machine learning systems rely on the *single representation trick* (Cohen and Feigenbaum, 1982) by representing objects as specially identified individual concepts. In many terminological knowledge representation systems (Nebel, 1990; MacGregor and Brill, 1992), instantiation is also computed as subsumption between a concept and an individual concept.

The θ-subsumption problem is known to be NP-complete (Garey and Johnson, 1979, problem LO18). Since any structured concept discovery algorithm will need to perform something closely related to subsumption checking, as we shall see below, the need for managing complexity arises. While structured concepts are very powerful, in order to be of practical use the average-case complexity of subsumption checking should be polynomial.

The complexity of subsumption can be managed in various ways, while retaining the spirit of structured concepts. One is to simply acknowledge cases where subsumption may take an inordinate amount of time to compute, relying on an operator to make guiding decisions. Another is to curtail the expressiveness of the hypothesis space, for example, by omitting background theories and requiring that all attributes in a structured concept occur only once. In such a case, there will only be one possible matching between any two concept variables, and subsumption can be computed in polynomial time. Another is to sacrifice completeness of subsumption checking, for example, by employing a beam search through possible θ-substitutions (Dietterich and Michalski, 1981). Another is to restrict the number of variables in a structured concept by aggregating groups of parts into lower-level objects. The knowledge representation community has two views on this tradeoff between expressiveness and tractability (Levesque and Brachman, 1987; Doyle and Patil, 1991). Some researchers assert that the expressiveness of a knowledge representation language (or machine learning hypothesis space) should not be compromised. Others argue that a hypothesis space should be restricted so that sound and complete inferences can be made in reasonable time.

2.4 Common subsumers and generalization

Machine learning systems rely extensively on computing a concept in a hypothesis space which subsumes two or more other concepts. For two concepts C and D, and theory $\mathcal{T}$, a *common subsumer* is a concept L such that $L \succeq_{\mathcal{T}} C$ and $L \succeq_{\mathcal{T}} D$. A *least common subsumer* has the additional property that for no other common subsumer $L' \not\equiv_{\mathcal{T}} L$ in the hypothesis space does $L \succeq_{\mathcal{T}} L'$.

Constructing a common subsumer between two structured concepts is straightforward and can be done in linear time: essentially any injective matching between parts of the concepts will build a common subsumer (Haussler, 1989). Computing a *least* common subsumer is difficult, and is readily shown to be isomorphic to the maximal matching problem which is known to be NP-complete (Garey and Johnson, 1979, problem GT50). See (Barrow and Burstall, 1976; McGregor, 1982) for further discussions of the maximal common subgraph problem.

A task closely related to the common subsumer problem is the *generalization* problem: given a theory $\mathcal{T}$ and a concept D, produce another concept C in the hypothesis space such that $C \succeq_{\mathcal{T}} D$. One rule of generalization, which incorporates a non-empty theory $\mathcal{T}$ and encompasses many of those generalization rules presented by (Dietterich and Michalski, 1981), is *truncation*: removing literals from saturated concepts (Rouveirol, 1994). For example, the concept $C1$ from Section 2.2.1 is saturated to the concept

$$\lambda\texttt{x.small(x)} \wedge \texttt{fluffy(x)} \wedge \texttt{dog(x} \wedge \texttt{pet(x)}.$$

A subsuming concept is then produced by removing one or more literals from this saturated concept.

As an aside, in the field of *inductive logic programming* (Muggleton, 1992) (see also Section 2.6.1 below), it is common to employ Plotkin's (1971) *least general generalization* algorithm as a common subsumer method. While the algorithm runs in polynomial time in the size of two input concepts, when used with a background theory it may produce very long expressions. These will likely contain many redundant variables and literals which aggravate subsequent subsumption computations. Removing these redundant literals requires expensive subsumption testing.

2.5 Concept classification and organization

Most structured concept discovery systems organize their discoveries into a network of concepts. A *concept taxonomy* is an acyclic directed graph with vertices denoting concepts and edges denoting subsumption, modulo transitivity and reflexivity. A new concept is *classified* by placing it in its correct location in the taxonomy, just below all *most specific subsumers*, and just above all *most general subsumees* (Woods, 1991). A concept C in the taxonomy is a most specific subsumer of a concept Q if $C \succeq_{\mathcal{T}} Q$, and for any other concept D in the taxonomy, $D \succeq_{\mathcal{T}} Q$ implies $D \succeq_{\mathcal{T}} C$. Similarly, a concept C in the taxonomy is a most general subsumee of a concept Q if $Q \succeq_{\mathcal{T}} C$, and for any other concept D in the taxonomy, $Q \succeq_{\mathcal{T}} D$ implies $C \succeq_{\mathcal{T}} D$.

To illustrate the theory outlined in this section, refer to Figure 2, which shows a taxonomy of structured concepts. A diagrammatic notation is used: two parts are related if and only if they are connected (on an adjacent row, column, or diagonal) in the diagram. For the two query concepts Q1 and Q2 at the right of Figure 2, the most specific subsumers (mss) and most general subsumees (mgs) are as follows:

$$\begin{aligned}
&\text{mss(Q1)} = \{\text{C6}, \text{C1}\}\\
&\text{mgs(Q1)} = \emptyset\\
&\text{mss(Q2)} = \{\text{C5}, \text{C7}\}\\
&\text{mgs(Q2)} = \{\text{C12}\}
\end{aligned}$$

The children or *immediate subsumees* (ims) of various concepts in the taxonomy are

$$\begin{aligned}
&\text{ims(C1)} = \{\text{C7}, \text{C9}\}\\
&\text{ims(C5)} = \{\text{C10}\}\\
&\text{ims(C12)} = \emptyset
\end{aligned}$$

The *fringe* of — all concepts subsumed by — various concepts in the taxonomy are

$$\begin{aligned}
&\text{fringe(C1)} = \{\text{C7}, \text{C9}, \text{C11}, \text{C12}, \text{C13}\}\\
&\text{fringe(C8)} = \{\text{C10}, \text{C11}, \text{C12}, \text{C13}\}\\
&\text{fringe(C12)} = \emptyset
\end{aligned}$$

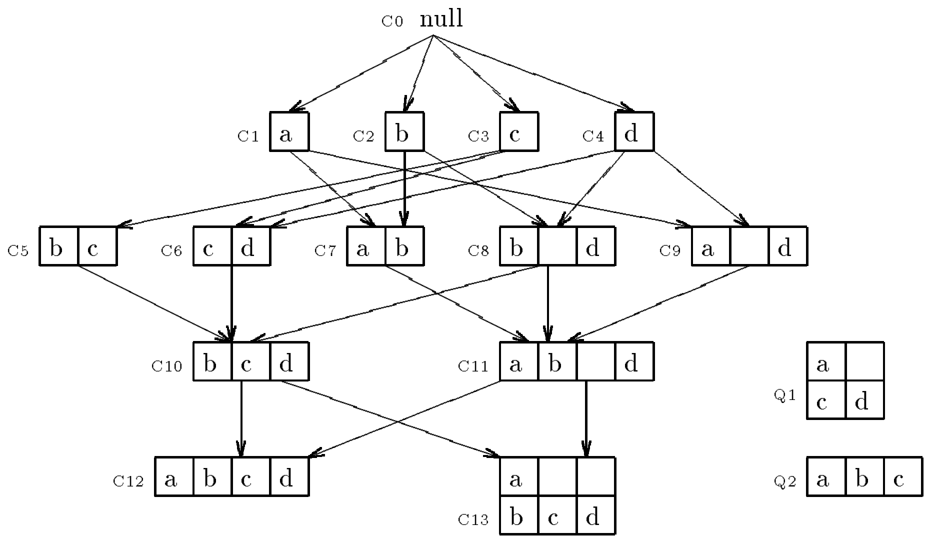


Figure 2: A structured concept taxonomy and two query structures.

Concept taxonomies can be used as background theories, as will be seen in a couple structured concept discovery systems below. For example, the taxonomy of Figure 2 is a representation of the theory

$$\begin{aligned}&\forall \mathtt{x}\, C12(\mathtt{x}) \Rightarrow C10(\mathtt{x})\\ &\forall \mathtt{x}\, C10(\mathtt{x}) \Rightarrow C6(\mathtt{x})\\ &\forall \mathtt{x}\, C9(\mathtt{x}) \Rightarrow C4(\mathtt{x})\\ &\forall \mathtt{x}\, C3(\mathtt{x}) \Rightarrow C0(\mathtt{x})\\ &\ldots\end{aligned}$$

2.6 Structured concept discovery

Concept discovery systems can be categorized into three groups, based on their degree of autonomy. *Concept learning* systems are "supervised" by a teacher and restricted by the assumption that all examples are covered by one concept. *Conceptual clustering* systems are "unsupervised" and autonomously partition a set of examples using multiple concepts. *Exploratory* approaches are not necessarily guided by examples, and must explore the incoming data or hypothesis space autonomously.

Although the sections below will introduce concept learning, the main focus of this paper will be on conceptual clustering of structured objects. A number of different systems for this task will be reviewed in Section 3.

2.6.1 Concept learning

The goal of *concept learning*, one of the most fundamental and well-studied machine learning tasks, can be stated very simply. Given a target concept and examples of that concept, form an intensional description of the concept. The inference from examples to concept description is inductive, rather than deductive.

More formally, consider a universe of discourse $\mathcal{D}$ and let $S \subseteq \mathcal{D}$ be the set of objects that belong to the target concept. Furthermore, let $E \subseteq S$ be a finite set of examples of the concept. Giving names $e_1, \ldots, e_n$ to these examples, axiomatize the world using a theory $\mathcal{T}$, and create a concept C such that for $i \in \{1, \ldots, n\}$

$$\mathcal{T} \models C(e_i). \tag{9}$$

Note that this expression can be computed by multiple proofs of instantiation relationships.

Concept learning from positive examples is an underconstrained task. Many concepts C will satisfy (9), including the two trivial concepts

$$\lambda \mathtt{x}.\mathtt{true},$$

and

$$\lambda \mathtt{x}.(\mathtt{x} = e_1) \vee \ldots \vee (\mathtt{x} = e_n).$$

(Note that this expression is only valid in a hypothesis space supporting disjunction and equality). This invokes the well-known *problem of induction*: there is no logical basis for choosing between two or more admissible concepts. The problem arises regardless of the choice of hypothesis space. In structured concept learning there have been several attempts to address this problem. The most common one is to use *negative examples* of the concept, insisting that no negative example is instantiated by the concept (Muggleton and Feng, 1992; Winston, 1975; Quinlan, 1990). Another technique is to define the hypothesis space such that the system is incapable of expressing unreasonable inductive choices. Another is to *minimally* generalize the examples E, for instance, by creating their least common subsumer (Dietterich and Michalski, 1981). Another is to use an extra-logical preference criterion over inductive hypotheses. For example, both Conklin and Witten (1994) and Muggleton et al. (1992) show how the tradeoff between theory simplicity and the fit of a theory to examples can be balanced using a minimal length encoding heuristic. Finally, some systems rely on an "oracle" or a user to ensure them that overgeneralization has not occurred (Sammut and Banerji, 1986).

There has been much work on structured concept learning. Dietterich and Michalski (1981) present a survey of early work, including Winston's (1975) landmark ARCH system. Researchers later realized that the hypothesis space of *logic programs* (Lloyd, 1987) is very rich, supports disjunctive concepts, and naturally unifies and supports structured concept learning (Shapiro, 1983; Buntine, 1988). The new field of inductive logic programming (Muggleton, 1992) has made great strides in formalizing the tasks of induction, generalization, and new predicate discovery in the presence of rich background theories expressed as logic programs.[7] The field is advancing rapidly: see (Aha, 1992) and (Muggleton, 1993) for reviews.

[7]Although systems such as FOIL (Quinlan, 1990) and GOLEM (Muggleton and Feng, 1992) allow only ground atoms as background knowledge.

2.6.2 Conceptual clustering

The task of concept learning, described in the previous section, is to discover a concept which is satisfied by a given set of examples. The task of *conceptual clustering* is to discover a *set* of concepts which, taken together, cover a set of examples. An early definition of conceptual clustering was set out by Fisher and Langley (1986): there are many slight variations on the task. Consider a universe of discourse $\mathcal{D}$ and let $E \subseteq \mathcal{D}$ be a finite set of examples. Giving names $e_1, \ldots, e_n$ to these examples, axiomatize the world using a theory $\mathcal{T}$, create a partitioning R of the examples, and for each $r \in R$ create a concept C such that for all $e \in r$

$$\mathcal{T} \models C(e). \tag{10}$$

A conceptual clustering system must therefore address two problems (Fisher and Langley, 1986). The *aggregation* problem creates useful partitions of the example set. The *characterization* problem — the inductive concept learning problem — creates disjoint concepts which cover these sets. Conceptual clustering systems have tackled the vast search space of clusterings in various ways. One is to define a preference ordering on clusterings. Another is to perform incremental hill-climbing over the space of clusterings, perhaps also using preference criteria (Lebowitz, 1987; Fisher, 1987). A clustering will depend in such systems on the order of example presentation. Another is to enforce the restriction that the partition should be disjoint, as is done by the CLUSTER/2 system (Michalski and Stepp, 1983b). In this case, instances of other concepts can be used as negative examples to a concept learning algorithm. Another approach is to fix the number of clusters in advance, and search over this much restricted hypothesis space. For example, the CLUSTER/2 system selects a fixed-size set of "seed" objects which are assumed to belong to disjoint clusters.

Conceptual clustering can be viewed as a restricted discovery process, as concepts represent recurrent structure among their instances. As we have seen, this recurrent structure can be put to many uses, such as indexing new structured objects with the same regularities, and making analogical inferences. When the exhaustiveness and disjointness restrictions of clustering are dropped or weakened, conceptual clustering is transformed into a more abstract discovery process: these are instances of *concept discovery* systems. Most work on conceptual clustering has focused on concepts expressed as attribute-value restrictions (disallowing any binary predicates). Recently there has been much interest in structured conceptual clustering.[8] The concept discovery systems reviewed in the next section all perform conceptual clustering of structured objects.

It is appropriate at this point to review a fundamental distinction between conceptual clustering (as developed by the field of machine learning) and numerical taxonomy (as developed by the field of pattern recognition). Numerical taxonomy methods represent objects as vectors in a multidimensional parameter space. A concept is also a vector representing a most representative instance or a centroid of a region of multidimensional space. Instantiation relations are not computed by model-theoretic proof, but rather by measuring the *similarity* between an instance to a concept. The distinction between numerical taxonomy and conceptual clustering is blurred in some systems. For example, as we shall see, some systems

[8]It appears that one of the first suggestions that conceptual clustering systems might use structured objects was given by Langley and Carbonell (1984).

Table 2: Pertinent questions for structured concept discovery systems.

What is the hypothesis space?
Can multilevel structured objects be represented?
What form, if any, of background knowledge is employed?
How is background knowledge used?
What is the learning algorithm? Is it incremental?
What is the complexity of subsumption testing and concept discovery?
What is the appropriate or intended semantics of concepts?
What form of memory structure is created by the system?
Are discovered concepts semantically or structurally disjoint?
How are new concepts created?
How are concepts selectively acquired?
Does the system monitor and evaluate its own performance?

use a model-theoretic definition of subsumption, but use a similarity semantics for instantiation. Other types of clustering can be characterized by the semantics given to concepts: for example, fuzzy clustering (Bezdek and Pal, 1992) and probabilistic clustering (Cheeseman et al., 1988). See (Pitt and Reinke, 1988; Gennari et al., 1989; Fisher and Langley, 1986; Michalski and Stepp, 1983a) for further comparisons of conceptual clustering and numerical taxonomy.

3 A review of selected methods

The preceding sections have laid the theoretical foundations for presenting a number of implemented systems which perform structured concept discovery. The order of presentation is roughly according to the first date of publication. Each system surveyed has different strengths, weaknesses, and features. Table 2 lists a number of questions that will be addressed throughout this section.

There has been a history in structured concept discovery research of modifying an attribute-value system to deal with structured descriptions. For example. CLUSTER/S is derived from CLUSTER/2, MERGE is derived from UNIMEM, LABYRINTH is derived from COBWEB. In the appropriate sections below a description of the associated attribute-value conceptual clustering system will be given.

3.1 MERGE

The UNIMEM system (Lebowitz, 1987), like COBWEB, addresses the problem of incremental conceptual clustering in an attribute-value hypothesis space. UNIMEM has an interesting evaluation scheme for concepts, based on a *predictability* analysis (Lebowitz, 1986b). Similar to COBWEB, this is based on how well inferences about missing features can be made using a discovered concept. For each feature (unary literal) in a concept, a *confidence* score says how many times that feature had been confirmed or disconfirmed by examples in the past.

After a new example passes through a concept — during the most specific subsumer computation — the confidence values are inspected. Features with low confidence are removed. If this results in a concept with too few features, the concept is removed. When an example reaches a most specific subsumer, high similarity with any of its immediate subsumees will trigger the creation of the least common subsumer[9], and the requisite local rearrangement of the concept taxonomy.

The MERGE system (Wasserman, 1985) extends the UNIMEM system to deal with multilevel structured objects and binary relations. The hypothesis space used is presented in (Wasserman and Lebowitz, 1983). Like UNIMEM, MERGE finds the most specific subsumers of an input example and, for each, computes the similarity between its immediate subsumees and the example. To test for similarity of two structured concepts, MERGE considers all possible matchings between their parts, and counts common literals for each matching. These common literals, for the maximal matching, can be used to build a common subsumer. The decomposition of an object into parts seems to be only for memory organization purposes, and it appears that the π relation is considered to be transitive, as the part decomposition is sometimes ignored when computing a structural match between two concepts. The LABYRINTH (Thompson and Langley, 1991) and IMEM (Conklin and Glasgow, 1992) systems, surveyed below, remedy this problem.

MERGE is closely related to UNIMEM, and therefore inherits the strengths and weaknesses of that system. Examples are instantiated according to *similarity* with concepts in the network, and the semantics of the system is largely operational. Whereas links in the concept taxonomy usually denote model-theoretic subsumption, it is not clear that the feature removal process preserves the integrity of the concept taxonomy. Like UNIMEM, MERGE cannot employ background knowledge. A strength of MERGE, like UNIMEM, is that concepts can be nondisjoint. This introduces flexibility into the clustering process in that multiple concepts can be used to describe the same example.

3.2 Levinson

Levinson (1985) describes a self-organizing information retrieval system for graphs. A database of graph structures is indexed using a taxonomy of discovered structured concepts. Subsumption is computed using a subgraph isomorphism test. The hypothesis space used — connected graphs with labelled edges and vertices — is equivalent to structured concepts with unary and binary predicates. Only one relation is allowed between any two parts. Negation can be expressed using special "negated" edges.[10] Multilevel objects, multinary relations, or background knowledge cannot be expressed, although recent research by Levinson (1994) corrects these problems.

The central tenet of Levinson's work is that a taxonomy provides an efficient indexing mechanism for structured objects. This indexing mechanism can be used to answer three

[9]In an attribute-value hypothesis space such as that used by UNIMEM, there is a unique least common subsumer for any two concepts.

[10]The absence of an edge in a graph does not mean that two parts are *not* related; if this were so, the subgraph isomorphism test for subsumption would be unsound, and a slightly modified definition of "subgraph" must be employed. See also the footnote to Figure 1.

types of queries. The *generalization* query asks, for some structured concept Q, what concepts are most specific subsumers of Q. The *specialization* query determines the most general subsumees of Q. A *close match* to Q is any concept subsumed by one of Q's most specific subsumers, that is, some concept in the fringe of some element of $\mathrm{mss}(Q)$. Conklin (1993a) outlines the properties of a similarity measure for which close match retrieval is sound with respect to similarity.

Levinson's learning algorithm is similar to that of UNIMEM (and MERGE). It is incremental, incorporating individuated structured objects one by one into an evolving concept taxonomy. For each close match of an input example, the system computes a maximal common subgraph (thereby a least common subsumer) and a minimal different subgraph.[11] A minimal different subgraph of two concepts is a most general generalization of one which does not subsume the other. The newly-formed concept is retained and classified (see Section 2.5) only if it may improve retrieval efficiency of objects already encountered. Since concepts added early on in an incremental stream of examples may later turn out to be detrimental to retrieval efficiency, the system has a concept "garbage collection" utility which removes undesirable concepts. Levinson's learning algorithm differs slightly from UNIMEM's in that the latter inspects only the immediate subsumees, while Levinson inspects the complete fringe of a most specific subsumer. Similarity is not used as a basis for discovered concept acquisition.

Levinson introduced some important ideas to structured concept discovery, including its formulation in terms of information retrieval, the restricted yet powerful hypothesis space of labelled graphs, and the criterion for retaining concepts. The approach relies extensively on maximal common subgraph calculation and subsumption computation. Although an efficient topological search is used to compute the most specific subsumers of an example, mechanisms for improving the average-case complexity of subsumption computation must exist. This problem will be revisited in Section 4 of this paper.

3.3 CLUSTER/S

As discussed earlier, the CLUSTER/2 system (Michalski and Stepp, 1983b) transforms the abstract conceptual clustering task into a series of supervised learning problems. Unlike all of the systems surveyed above, CLUSTER/2 uses background knowledge. This is expressed as a network of determination rules between attributes. Examples are not completely saturated with the background knowledge, rather, the network directs the partial saturation process by inspecting the set of attributes present in an example. The intuition behind this restricted saturation process is that not all background knowledge will be relevant to a given clustering task (Fisher and Pazzani, 1991). CLUSTER/2 uses a complex evaluation function to drive its search over clusterings: this is roughly based on the simplicity of cluster descriptions and the fit of the clustering to the examples.

[11]By definition, a *maximal* common subgraph is not a subgraph of any other subgraph. A *maximum* common subgraph has more vertices than any other maximal common subgraph. Levinson uses a heuristic common subgraph algorithm which does not guarantee the computation of either a maximal or a maximum common subgraph (Levinson, personal correspondence). It is also not clear what the learning algorithm does when more than one maximal common subgraph exists.

The CLUSTER/S (Stepp and Michalski, 1986) system clusters structured objects, and invokes CLUSTER/2 as a subroutine. Examples are preprocessed by CLUSTER/S — translated into an attribute-value language — where CLUSTER/2 can apply. Essentially this involves finding a common subsumer L of *all* structured examples (using a heuristic beam search), and freezing a subset of variables in each example by substituting unique constants for them. Some variables, not participating in L but connected by relations to variables in L, are also frozen. The resulting variable-free descriptions are then processed by the CLUSTER/2 system. A postprocessing step melts the discovered concepts, substituting variables for constants and converting them back into a structured concept representation.

Despite the important advances made by CLUSTER/S as one of the first structured concept discovery systems to incorporate background knowledge, it has a shortcoming. Since only *one* least common subsumer of all examples is computed, the system cannot be applied to problems where there is no single, meaningful common subsumer. Although CLUSTER/S uses a hypothesis space capable of representing taxonomies of structured concepts, its discovery algorithm is limited.

3.4 KBG

Like MERGE, KBG (Knowledge Base Generalizer) (Bisson, 1992a) uses a similarity valuation to provoke concept discovery. Highly similar examples are generalized. An example in KBG is a subset of a database of ground atomic facts. Negated literals cannot be expressed. Background knowledge is applied to each example by a complete saturation process. Unlike CLUSTER/S, which controls saturation, KBG computes a deductive fixpoint of each example. Following this process, each example is individuated.[12]

The uniqueness of the KBG learning algorithm is in its computation of similarity between two examples. Unlike MERGE, KBG does not consider all possible matchings between structured concepts. The similarity measure (Bisson, 1992b) considers, for each pair of objects in an example, the number of common predicates in which the pair occur as the same argument. Partial relational similarity is taken into account by inductively considering the similarity of other pairs of entities in each common predicate. The result of a similarity computation is a valuation between the two examples, and a valuation between each pair of objects in the examples. The similarity computation is polynomial in the number of literals in the examples.

The learning algorithm of KBG operates as follows. For a set E of examples, KBG builds a square similarity matrix from all similarity valuation computations. Following this process, the system chooses the two[13] most similar examples A and B of E and, using the injective matching derived by the similarity computation, computes a common subsumer G (not necessarily a least common subsumer). The clustering process iterates with the new set $E \Leftrightarrow \{A, B\} + \{G\}$ of "examples", and terminates when this set is a singleton. The resulting concept taxonomy is incomplete: some subsumption relationships are not present. This is

[12]Bisson refers to "exemplifying of a generalization"; the inverse of individuation. To remain in line with the theoretical foundations section, I prefer to view KBG as working with individuated examples.

[13]KBG will consider common subsumers of more than two examples using a *thresholding* method, but for simplicity here I shall only consider the two-example case. Thresholding also allows discovered concepts to be nondisjoint.

because discovered concepts are not classified back into the concept taxonomy. To correct this, a postprocessing step determines subsumption relationships missed during the learning algorithm (Bisson, personal correspondence). This would appear to be an expensive step, since it involves subsumption computations. Note that the learning algorithm is otherwise quite efficient, since least common subsumer and subsumption computations are avoided. The price paid for this efficiency is that overly-general concepts may be created.

3.5 LABYRINTH

The LABYRINTH system (Thompson and Langley, 1991) is based on the COBWEB conceptual clustering system (Fisher, 1987). COBWEB manipulates *probabilistic* concepts. These concepts encompass a set of features, along with a conditional probability of the feature given membership in the concept. Although COBWEB creates a concept hierarchy, links between concepts do not represent logical subsumption. The goal of COBWEB is to create concepts that maximize the expected number of features that can be correctly predicted for instances of the concept. COBWEB passes examples through its hierarchy — a process analogous to the computation of a most specific subsumer (Section 2.5) — potentially creating new concepts in the process. Concept discovery is guided by a measure which attempts to maximize intracluster similarity and intercluster dissimilarity. When an example is sorted to a node in the hierarchy, it may recursively pass through that node, or trigger concept formation at that level. Two other learning operators are employed to compensate for incremental learning and example presentation order effects: these attempt to correct a suboptimal hierarchy by merging or splitting concepts at a level of the hierarchy.

Concepts in COBWEB are probabilistic, and there is no use for model-theoretic subsumption. This means that the results of discovery can be difficult to interpret (Langley, personal correspondence). All concepts in COBWEB are "disjoint", however, unlike model-theoretic systems, where disjointness is a *semantic* property of the discovered concepts (recall Section 2.1), in COBWEB disjointness is a purely *structural* property of a particular concept taxonomy.

LABYRINTH extends the weak hypothesis space of COBWEB, and is capable of expressing multilevel structured objects with binary relations between parts. Of the systems surveyed earlier which use background knowledge, none were "constructive": able to use named discovered concepts as new unary predicates in the first-order language. The LABYRINTH, SUBDUE, and IMEM systems incorporate this important idea. LABYRINTH can express negated literals. A concept with n parts will have n probability distributions for the labels of all possible component concepts. For every binary predicate symbol there will be an additional $O(n^2)$ entries in the concept.

LABYRINTH recursively incorporates an individuated structured object in a depth-first fashion; for a level-1 concept this entails classifying each level-0 part, obtaining a level-0 label for each part. For all possible matchings of labels to concept parts, LABYRINTH evaluates the score of the match. This matching process is performed with each child of a node; this determines the roles of parts so that COBWEB's category utility measure can be applied. As an object progresses through the taxonomy, LABYRINTH considers generalizing the features of a node in the taxonomy. This involves computing a common subsumer of two corresponding features and evaluating the resulting concept using an equation designed to avoid overgeneralization. Unlike CLUSTER/S and KBG which preprocess examples using saturation, in LABYRINTH background knowledge is applied during concept formation. Previously discovered concepts can be used as background knowledge. LABYRINTH has some problems that prohibit its use on real applications. The time complexity introduced by its exhaustive matching routine will be unacceptable in domains with large structured objects.

3.6 SUBDUE

The systems presented so far have performed generalization from two or more examples. By contrast, the SUBDUE system of Cook and Holder (1994) analyzes a single structured object to detect recurrent connected subgraphs. The hypothesis space appears to be equivalent to that of Levinson's system, although an "inexact" subgraph isomorphism test is used to validate subsumption relationships. A minimal length encoding measure is used to quantify the "interestingness" of a discovered structured concept. This measure coaxes the system away from concepts which are too general or too specific. Overly general concepts will require much additional information to reconstruct the original graph, while overly specific concepts will be too complex. In neither case will the original structured object be compressed well by the discovered concept.

The fundamental limitation of this method is that, by definition, substructures which do not recur within a *single* structured object will not offer any compression and will not be proposed by SUBDUE. Although this is an unrealistic restriction for general machine learning, future work could profitably combine inter-example methods with SUBDUE's intra-example discovery method.

In SUBDUE, the components of a concept can refer to a previously discovered concept. Thus, the system is capable of building a concept taxonomy of multilevel structured concepts. Once a substructure is identified in an example, that example is "parsed" and re-expressed using that substructure. This is in contrast to the LABYRINTH and IMEM systems which assume that examples are parsed in compatible ways. The parsing process of SUBDUE appears to have unspecified semantic implications, since new π relations, and new predicates expressing relations between higher-level objects must be introduced into the underlying first-order language.

3.7 IMEM

The IMEM (Image MEMory) system (Conklin and Glasgow, 1992) incorporates aspects of both LABYRINTH and Levinson's system. Like LABYRINTH, it can represent multilevel structured concepts, negated literals, and can use the names of discovered concepts as new unary predicates. The learning algorithm of IMEM is similar to Levinson's, and it also uses a model-theoretic semantics for concepts. Unlike both systems, IMEM can express multinary relations between parts.

The knowledge representation scheme used by IMEM is somewhat unique. It uses a syntax called an *image term* to represent structured concept descriptions. An image term is formed by associating an image (a set of concept/coordinate pairs) with a set of relation identifiers. To each relation identifier there corresponds a partial Boolean-valued function mapping a tuple of image components to values. Relations are computed directly from the image using functions. The space complexity of a structured concept description is therefore linear rather than polynomial in the number of concept variables. Implicitly present in a structured concept description, for a given relation, are *all* tuples for which the associated partial function is defined. The drawback of this complete representation is that only coarse-grained truncation is available as a generalization operator. IMEM uses two truncation operators: removing a part from an image (i.e., removing all literals that involve a given variable), or removal of a relation from a relation set (i.e., removing all literals with a given predicate symbol). Although a complete representation is weaker than an arbitrary graph representation, as used by Levinson for example, it is comprehensible and has been shown to be surprisingly effective in characterizing many different types of spatial concepts. It should be noted that the subsumption problem in this representation remains NP-complete.

IMEM uses background knowledge in the form of a concept taxonomy. This can be viewed as an attribute hierarchy, but like LABYRINTH, discovered concepts also constitute dynamic background knowledge. This knowledge is applied after a maximal structural match. This generalization operator is called “part replacement” as it replaces a component of an image with a more general one.

Of the systems surveyed IMEM is most closely related to Levinson’s. The learning algorithm differs slightly in that IMEM, after computing the most specific subsumers of an example, computes a common subsumer with only one most similar concept. IMEM, unlike Levinson’s system, does not compute “greatest different” subsumers. IMEM was designed as a self-organizing information retrieval system for images. It supports generalization, specialization, and close match queries. Like Levinson’s system, the concept taxonomy formed is sound and complete with respect to subsumption. Whereas Levinson uses a heuristic designed to speed graph retrieval, IMEM uses a heuristic designed to maximize intercluster dissimilarity and intracluster similarity. These measures are in line with classical information retrieval goals of balancing recall and precision (Salton and McGill, 1983).

3.8 Miscellaneous systems

This paper has surveyed several major structured concept discovery systems in detail. Each system represented some advancement over previous systems in the technology of structured concept formation. Several general-purpose discovery systems were omitted. Although secondary to its main goal of concept learning, Winston’s (1975) ARCH system includes a “grouping” method which discovers concepts comprising, for example, sequences of parts related by a common relation. These structured concepts are used to deepen the hierarchical description of an object.

The RINCON system (Wogulis and Langley, 1989) attempts to improve the efficiency of the relational concept instantiation problem by creating and retaining “intermediate” concepts. The DMP system (Parsons, 1989) discovers structured concepts within a database of ground facts. The CAFE system of Handa et al. (1994) has made minor extensions to LABYRINTH’s hypothesis space and concept evaluation function. It allows the parts of concepts to be "context-sensitive" by including a description of the inverse of the π relation. Mineau and Godin (1994) describe a conceptual clustering system for restricted structured concepts. In contrast to other systems, their system creates a *complete* concept lattice: every subsumer of the examples is formed and made explicit. The IDS system (Nordhausen and Langley, 1990) has a structured concept discovery component very similar to MERGE. Yoshida et al. (1993) describe CLIP, an intra-example discovery system similar to the SUBDUE system.

There has been a longstanding interest in the computer vision and pattern recognition communities in the use of structured object representations and relational models. Structured objects and concepts are a powerful way to describe the necessary and sufficient features and relations of a visual object. Most of the machine learning work in this area has been in supervised concept learning (Winston, 1975; Connell and Brady, 1987), although researchers have also considered the problem of organizing relational models for efficient search. A common solution is to perform a numerical clustering procedure on a database of structured objects (Sengupta and Boyer, 1993; Shapiro and Haralick, 1982; Segen, 1990). Structured concepts will then have a similarity semantics, and links in the cluster taxonomy will not necessarily reflect subsumption. Dey et al. (1993), by contrast, do use a model-theoretic notion of subsumption. Conklin (1993b) shows how the IMEM system applies to the relational model organization problem. In a somewhat related field, research in retrieval of structured pictures from image databases has used a concept abstraction or generalization operator (Chang and Liu, 1984).

The notion of a structured concept reappears frequently in many different problem domains. The structural chemistry community, for example, has had a longstanding interest in graph structures. An early system, Meta-DENDRAL (Cohen and Feigenbaum, 1982), created a taxonomy of molecular graphs for use in mass spectrum analysis. The system of Okada and Wipke (1989) indexes a database of molecular graphs using a taxonomy of discovered maximal common subgraphs. Okada (1993) shows how this taxonomy can be used for analogical inference. Structurally similar molecules are retrieved by a most specific subsumer computation, and properties of molecules are inferred from similar instances. Wilcox and Levinson (1986) show how Levinson's system can be applied to a database of molecular structures for efficient information retrieval. LABYRINTH has been applied to the problem of DNA promoter sequence recognition (Thompson et al., 1991). IMEM has been applied to the discovery of small molecule conformation classes (Conklin et al., 1993), and to protein motif discovery (Conklin et al., 1994).

3.9 Discussion

The previous section has surveyed a number of structured concept discovery systems. These were reviewed with particular attention to the *knowledge level*, that is, to the expressive power of the hypothesis space and the meaning of concept subsumption and not to the specific symbol structures used. Table 3 presents these systems according to several dimensions. The "semantics" column refers to the underlying semantic theory of a concept. In the case of probabilistic concepts (LABYRINTH), concepts are interpreted as conditional probability distributions over literals. This interpretation has the undesirable property of

Table 3: A comparison of several structured concept discovery systems. Incr: incremental, Nd: nondisjoint, SA: selective acquisition method.

Name	Relations	Level	Theory	Semantics	Incr	Nd	SA
MERGE (1985)	binary	any	no	combined	yes	yes	similarity
Levinson (1985)	binary	1	no	model	yes	yes	cost
CLUSTER/S (1986)	multinary	n/a	yes	model	no	no	seed
KBG (1992)	multinary	n/a	yes	model	no	yes	similarity
LABYRINTH (1991)	binary	any	yes	probability	yes	no	probability
SUBDUE (1994)	binary	any	no	similarity	no	yes	compression
IMEM (1992)	multinary	any	yes	model	yes	yes	similarity

not supporting a clear definition of subsumption. In the case of model-theoretic semantics, the most common form, concepts have an extension (recall Section 2), and subsumption is defined as a subset relation between extensions in all possible interpretations of the concept. In these systems, unlike probabilistic systems, links between concepts in a taxonomy have a sound denotation with respect to subsumption. Finally, the MERGE system uses a confusing *combined* semantics, where links in the concept taxonomy may denote model-theoretic subsumption, but instantiation is performed using a similarity comparison with a concept. The reasons for this unfortunate development stemmed from the early UNIMEM system, which was more concerned with structural memory organization than with sound semantic principles.

The “Relations”, “Level”, and “Theory” columns of Table 3 attempt to give an idea of the expressive power of a hypothesis space. For example, the LABYRINTH system employs background knowledge, supports multilevel structured concepts, and allows only binary relations between parts to be expressed. Multinary relations can be expressed as compositions of binary relations, but this becomes tedious with large structured objects. The CLUSTER/S and KBG systems do not represent structured objects as multilevel objects: rather, they support arbitrary conjunctive expressions. While their hypothesis space is strictly more expressive than that of other systems, they will suffer from severe computational difficulties when subsumption between large structured concepts must be performed. In systems which employ a true multilevel organization for structured concepts, objects can be organized so that a less expensive subsumption computation is faced (recall Section 2.3).

The “Nd” column of Table 3 states whether the classes produced by the discovery algorithm are nondisjoint. As stated earlier, the ability to produce nondisjoint categories is one of the differences between a conceptual clustering and a more flexible concept discovery system. In the two systems that produce disjoint classes, a distinction must be made as to whether classes are *structurally* disjoint (LABRYRINTH), or whether they are also *semantically* disjoint (CLUSTER/S).

The “Incr” column of Table 3 denotes whether learning algorithm is incremental. Nonincremental systems, such as CLUSTER/S and KBG, will have serious difficulties in coping with large numbers of examples. Extending CLUSTER/S to be incremental would be difficult due to its semantic disjointness requirement on discovered concepts. As pointed out by

Bisson (1992b), KBG would be fairly easy to coax into an incremental mode.

Finally, the "Sa" column of Table 3 refers to the manner in which concepts are selectively acquired (Markovitch and Scott, 1993) or discovered. LABYRINTH is derived from COBWEB and uses a predictiveness measure to judge the expected worth of a potential concept. Concepts formed by the CLUSTER/S system wholly depend on the initial "seed" objects selected from the pool of examples. Levinson (1985) employs a sophisticated heuristic which measures whether the new concept will lead to more efficient retrieval of existing examples. The most common selective acquisition mechanism is to create a concept when a new example is *similar* to a previous example. This similarity function can be purely structural (IMEM), based also on similarity of unary attributes (MERGE), or can be pseudo-structural (KBG).

4 Conclusions

This paper has presented a unified theory of structured concept discovery, and has discussed in detail several operational machine learning systems in this paradigm. Strengths and weaknesses of each system were outlined. This section will present some potentially important future research directions for the field. Haussler (1989) attempted a similar task; however, since that paper was mainly concerned with hypothesis space theory and not with operational systems, several issues were not anticipated. Below I will critically review a few of Haussler's proposals, focusing on if and how they have been addressed by current state-of-the-art systems. I will then propose my own new set of critical future research directions.

One of Haussler's proposals was to replace the $\exists^*$ quantifier (Section 3.2) by the $\exists$ quantifier. This has the effect that a part of a structured object can instantiate more than one variable of a structured concept. In computer vision terminology, this means considering a *relational homomorphism* rather than a *relational monomorphism* between concept variables (Vosselman, 1992; Haralick and Shapiro, 1993). In terms of θ-subsumption computation, it means searching over the much larger space of both injective and non-injective substitutions. This tradeoff with added search complexity must be justified with convincing examples. In computer vision and chemistry applications, for example, monomorphisms are commonplace. None of the systems surveyed above use general relational homomorphisms.

A second proposal of Haussler is to increase hypothesis space power by including disjunction as a basic operator. Although this sounds simple enough, there are several ways disjunction might be incorporated. One is to allow arbitrary disjunctive normal form expressions as concept descriptions. As mentioned earlier, the field of inductive logic programming uses such a hypothesis space. A simpler idea is to allow disjunction among unary predicates over the same variable. Something similar to disjunction can already be expressed by any conjunctive hypothesis space using background knowledge. A common subsumer of two attributes will include in its extension all objects with either attribute. This mechanism is not quite equivalent to disjunction as the common subsumer may be more general than the disjunction of the two attributes. This points to a problem when incorporating proper attribute disjunction: when generalization is to be performed, the least common subsumer can tend to be overly complex and specific. It is necessary to introduce additional preference criteria for candidate generalizations. It appears that this is a classical complexity versus fit problem, and Conklin and Witten's (1994) complexity-based induction approach may be useful here.

Another topic that came up in Haussler's paper was relation subsumption. For example, the relation "close" subsumes the relation "very close". In chemistry, bonds of lower orders can be viewed as subsuming those of higher orders (Okada and Wipke, 1989). The use of relation subsumption is also related to the use of more powerful types of background knowledge, e.g., definite clause programs as used by the inductive logic programming community. The KBG system uses a taxonomy of relations that is applied during the saturation process.

Finally, Haussler refers to the need to consider multilevel structured objects. Clearly this proposal has been satisfied and implemented in the LABYRINTH and the IMEM structured concept discovery systems.

I shall divide my proposals into two types; those concerned with hypothesis space power and those concerned with memory organization, retrieval, and system issues.

A recurrent issue in knowledge representation is finding a good *ontology*; predicates used to describe objects in the world. Structured concept discovery systems face the same problems. For example, personal experience with the IMEM system has indicated that "correct" concepts emerge almost immediately from examples when the appropriate set of relations are used. This indicates that a topic of great importance is the investigation of *constructive* learning, whereby a concept discovery system has the ability to propose, represent, and evaluate new relations to be used the first-order language. Crandell and Smith (1983), for example, do a relational clustering to discover an ontology for molecules. The relational discovery task is difficult because of the enormous space of new relations that can be constructed from an initial language. On a brighter note, however, all structured concept discovery systems perform a constructive discovery of concepts — essentially unary relations — and it seems like much of the theory is already in place to cluster relational examples.

An intriguing extension to a hypothesis space is to allow for *cyclic* concepts, where the name of a concept reappears in its description. Consider, for example, the concept of a crystal, which is a basic structural motif repeated ad infinitum. Such a concept could have the form

$$X \stackrel{\text{def}}{=} M \vee \ldots X \ldots$$

where M is the basic motif. The field of inductive logic programming has already investigated recursive relational concepts; many of these principles could apply to structured concept discovery. Cyclic concepts are also studied by the knowledge representation community (Nebel, 1991). In pragmatic terms, a cyclic structured concept would be a compact representation of what would otherwise require arbitrarily many separate concepts of various specificity. None of the surveyed systems purportedly allow for cyclic definitions, in fact, few (e.g., IMEM, LABYRINTH) allow for a discovered concept to be expressed in terms of another previously discovered one.

The computational advantages of using a multilevel description have been pointed out in this paper. However, penalties are incurred by moving to a multilevel representation. First, there is no "correct", and sometimes no intuitive, way to decompose an object into subgroups. This is reminiscent of the general segmentation problem of computer vision and the "perceptual grouping" task (Marr, 1982; Pentland, 1986). Multilevel concept instantiation will require that a structured object be parsed in a way compatible with the concept.

Also, when an object is parsed, "cousin" relationships are awkward to maintain. The latter problem is a necessary tradeoff that must be accepted; the former could be addressed by a rigorous and extended theory for the π relation.

A topic that must be explored in more detail is the evaluation of discovered concepts during selective acquisition and retention. Recall that a selective acquisition method decides, at the time a concept is created, whether it should be incorporated into the background knowledge. A selective retention mechanism decides, in the future, whether to discard a concept that does not appear to be "useful". Thus any scheme for evaluation concepts must be based on a theory of pragmatics or learning performance. For the concept learning task, the goal is to improve the accuracy of classification. A selective acquisition measure should aim to learn concepts that may have higher accuracy, whereas a selective retention mechanism would discard concept descriptions with low accuracy.

For general concept discovery, unfortunately, evaluation is not so clear. Many different, and equally valid, retention measures can be proposed. One can wish that known concepts are "rediscovered", or that discovered concepts are somehow interesting, unanticipated, or intuitively plausible to a human expert. This was the evaluation mechanism used by Conklin et al. (1993) in their discovery of pyranose molecule configurations. A stronger, less subjective criterion is employed by Levinson's system, where a discovered concept must improve retrieval efficiency. LABYRINTH tries to maximize the inferential predictiveness of features. IMEM aims for a balanced concept taxonomy for information retrieval purposes. The CLUSTER/S system employs a detailed evaluation function for proposed clusterings, but is not based on any theory of pragmatics. To summarize this discussion, there is no "correct" concept evaluation criterion, and this points perhaps for a need for a stronger theory of pragmatics for concept discovery.

The final future research topic I shall discuss is that of knowledge base organization and structured concept retrieval as borne out by the concept classification process (Section 2.5). There is no reason to suppose that the number of discovered structured concepts in an application will be small, and if many or all of these are to be retained it is necessary to employ efficient implementation techniques. It seems clear that the size of the concept descriptions must be kept to a minimum. Lebowitz' (1986a) idea is to employ "inheritance" in the frame or object-oriented programming sense. For propositional calculus and frame-based representations, inheritance is deterministic. For structured concepts it is necessary to specify, in the form of a graph embedding, how the parent and child concept are structurally related. Ellis (1991) offers an intriguing approach, which is to specify a sequence of transformation rules which rewrites the parent concept into equivalence with the child. In both of these approaches, in a deep concept taxonomy much of the structure of lower concepts can be "inherited" from above, perhaps resulting in a significant savings in memory.

Levinson (1992) presents a technique for reducing the average-case complexity of structured concept subsumption checking during classification. The technique assigns properties to nodes in a graph, thereby making them more specific and restricting the number of possible concepts they can subsume. A taxonomy of these "node descriptors" is used to index database graphs. Soundness of the technique can only be guaranteed by an subsequent expensive subsumption test, but the algorithm avoids this test. A similar method has been used in chemical structure retrieval systems (Nagy et al., 1988) which must avoid as many subgraph isomorphism tests as possible.

Conklin and Jenkins (1994) observe that a concept taxonomy, in a standard terminological logic (Nebel, 1990), can be viewed formally as a mathematical category, where the nodes of the category are canonical-form terms and the arrows are injective subsumption mappings. A speedup theorem of the paper is that the presence of the cached mappings determines subsumption relationships between concepts in the taxonomy and a new query concept. This theorem does not lead to a reduction in storage complexity, but rather a reduction in the number of primitive subsumption tests. Barrow et al. (1972) also make a category-theoretic connection between subsumption and relational structures, and present a similar speedup theorem. The authors state that "the remaining problem of this approach is that of setting up the modules [concepts] and network [concept taxonomy] automatically in a 'learning phase'." Clearly, this "remaining problem" can now be addressed by the new generation of structured concept discovery systems. Levinson's (1994) "Method V" retrieval algorithm uses a very similar arrow caching technique. All of these methods are fundamentally different from Ellis' (1991) technique, as the category theorem states that *all* injective mappings between parent and child concept must be cached, whereas Ellis' approach only stores one set of transformations.

Finally, although the picture is improving with recent forays into chemistry, vision, and database applications, there are many other interesting domains where data is described by structured objects and the field must be acutely concerned with moving away from toy "blocks-world" problems. Concept discovery systems must be able to deal with very large amounts of data. This requires a real concern for efficient implementation techniques. There is much research to be done, but the future looks bright and it should be possible to retain the high expressive power and generality of structured concept discovery systems while achieving results in important specific applications.

5 Acknowledgements

I would like to thank Gilles Bisson, Diane Cook, Bob Levinson, and Kevin Thompson for their very helpful comments on an earlier draft of this paper. This research has been supported by a Postgraduate Scholarship from the Natural Science and Engineering Research Council of Canada.

References

[Aha, 1992] Aha, D. 1992. Relating relational learning algorithms. In Muggleton, S., editor, *Inductive Logic Programming.* Academic Press. 233–260.

[Barrow et al., 1972] Barrow, H. G.; Ambler, A. P.; and Burstall, R. M. 1972. Some techniques for recognising structure in pictures. In Watanabe, S., editor, *Frontiers of Pattern Recognition.* Academic Press. 1–29.

[Barrow and Burstall, 1976] Barrow, H. G. and Burstall, R. M. 1976. Subgraph isomorphism, matching relational structures and maximal cliques. *Information Processing Letters* 4(4):83–84.

[Bezdek and Pal, 1992] Bezdek, J. C. and Pal, S. K. 1992. *Fuzzy models for pattern recognition: methods that search for structures in data.* IEEE Press.

[Bisson, 1992a] Bisson, G. 1992a. Conceptual clustering in a first order logic representation. In Neumann, B., editor, *Proc. ECAI–92: Tenth European Conference on Artificial Intelligence.* John Wiley and Sons. 458–462.

[Bisson, 1992b] Bisson, G. 1992b. Learning in FOL with a similarity measure. In *Proc. AAAI-92.* The MIT Press. 82–87.

[Buntine, 1988] Buntine, W. 1988. Generalized subsumption and its application to induction and redundancy. *Artificial Intelligence* 36:149–176.

[Chang and Liu, 1984] Chang, S. K. and Liu, S. Y. 1984. Picture indexing and abstraction techniques for pictorial databases. *IEEE Trans. Pattern Analysis and Machine Intelligence* 6(4):475–484.

[Cheeseman et al., 1988] Cheeseman, P.; Kelly, J.; Self, M.; Stutz, J.; Taylor, W.; and Freeman, D. 1988. Autoclass: A Bayesian classification system. In *Proceedings of the Fifth International Workshop on Machine Learning.*

[Cohen and Feigenbaum, 1982] Cohen, P. R. and Feigenbaum, E. A. 1982. *The Handbook of Artificial Intelligence*, volume 3. William Kaufmann.

[Conklin et al., 1993] Conklin, D.; Fortier, S.; and Glasgow, J. 1993. Knowledge discovery in molecular databases. *IEEE Trans. Knowledge and Data Engineering* 5(6):985–987. Special Issue on Learning and Discovery in Knowledge-Based Databases.

[Conklin et al., 1994] Conklin, D.; Fortier, S.; and Glasgow, J. 1994. Knowledge discovery of multilevel protein motifs. In Altman, R.; Brutlag, D.; Karp, P.; Lathrop, R.; and Searls, D., editors, *Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology.* AAAI Press.

[Conklin and Glasgow, 1992] Conklin, D. and Glasgow, J. 1992. Spatial analogy and subsumption. In Sleeman, D. and Edwards, P., editors, *Machine Learning: Proceedings of the Ninth International Conference (ML92).* Morgan Kaufmann. 111–116.

[Conklin and Jenkins, 1994] Conklin, D. and Jenkins, M. A. 1994. Compilation of description logics. Unpublished research report, in preparation.

[Conklin and Witten, 1994] Conklin, D. and Witten, I. H. 1994. Complexity–based induction. *Machine Learning* 16(3).

[Conklin, 1993a] Conklin, D. 1993a. Machine discovery of protein motifs. Submitted for publication.

[Conklin, 1993b] Conklin, D. 1993b. Transformation-invariant indexing and machine discovery for computer vision. In Bowyer, K. and Hall, L., editors, *Machine Learning in Computer Vision: What, Why, and How?*, number FSS-93-04 in AAAI 1993 Fall Symposium Series. 10–14.

[Connell and Brady, 1987] Connell, J. and Brady, M. 1987. Generating and generalising models of visual objects. *Artificial Intelligence* 31:159–183.

[Cook and Holder, 1994] Cook, D. J. and Holder, L. B. 1994. Substructure discovery using minimum description length and background knowledge. *Journal of Artificial Intelligence Research* 1:231–255.

[Crandell and Smith, 1983] Crandell, C. W. and Smith, D. H. 1983. Computer-assisted examination of compounds for common three-dimensional substructures. *J. Chem. Inf. Comp. Sci.* 23:186–197.

[Dey et al., 1993] Dey, L.; Das, P.; and Chaudhury, S. 1993. Recognition and learning of unknown objects in a hierarchical knowledge base. In Bowyer, K. and Hall, L., editors, *Machine Learning in Computer Vision: What, Why, and How?*, number FSS-93-04 in AAAI 1993 Fall Symposium Series. 15–19.

[Dietterich and Michalski, 1981] Dietterich, T. G. and Michalski, R. S. 1981. Inductive learning of structural descriptions. *Artificial Intelligence* 16:257–294.

[Doyle and Patil, 1991] Doyle, J. and Patil, R. S. 1991. Two theses of knowledge representation: Language restrictions, taxonomic classification, and the utility of representation services. *Artificial Intelligence* 48(3):261–297.

[Ellis, 1991] Ellis, G. 1991. Compiling conceptual graphs. In *Processing Declarative Knowledge '91 : Lecture Notes in Artificial Intelligence 567.* Springer-Verlag. 41–55.

[Fisher and Langley, 1986] Fisher, D. and Langley, P. 1986. Conceptual clustering and its relation to numerical taxonomy. In Gale, W. A., editor, *Artificial Intelligence and Statistics.* Addison-Wesley. chapter 4, 77–116.

[Fisher and Pazzani, 1991] Fisher, D. H. and Pazzani, M. 1991. Theory-guided concept formation. In Fisher, D. H.; Pazzani, M.; and Langley, P., editors, *Concept Formation: Knowledge and Experience in Unsupervised Learning.* Morgan Kaufmann. 165–177.

[Fisher, 1987] Fisher, D. H. 1987. Knowledge acquisition via incremental conceptual clustering. *Machine Learning* 2:139–172.

[Garey and Johnson, 1979] Garey, M. R. and Johnson, D. S. 1979. *Computers and intractability: A guide to the theory of NP-completeness.* W. H. Freeman.

[Genesereth and Nilsson, 1988] Genesereth, M. R. and Nilsson, N. J. 1988. *Logical Foundations of Artificial Intelligence.* Morgan Kaufmann.

[Gennari et al., 1989] Gennari, J. H.; Langley, P.; and Fisher, D. 1989. Models of incremental concept formation. *Artificial Intelligence* 40:11–61.

[Gottlob, 1987] Gottlob, G. 1987. Subsumption and implication. *Information Processing Letters* 24:109–111.

[Handa et al., 1994] Handa, K.; Nishikimi, M.; and Matsubara, H. 1994. Utilizing structure information in concept formation. *Machine Intelligence 13.*

[Haralick and Shapiro, 1993] Haralick, R. M. and Shapiro, L. G. 1993. *Computer and Robot Vision*, volume 2. Addison-Wesley.

[Haussler, 1989] Haussler, D. 1989. Learning conjunctive concepts in structural domains. *Machine Learning* 4:7–40.

[Hayes, 1985] Hayes, P. J. 1985. The logic of frames. In Brachman, R. J. and Levesque, H. J., editors, *Readings in knowledge representation.* Morgan Kaufmann.

[Langley and Carbonell, 1984] Langley, P. and Carbonell, 1984. Approaches to machine learning. *J. American Society for Information Science* 35(5):306–316.

[Lebowitz, 1986a] Lebowitz, M. 1986a. Concept learning in a rich input domain: generalization-based memory. In Michalski, R.; Carbonell, J.; and Mitchell, T., editors, *Machine Learning: An Artificial Intelligence Approach*, volume II. Morgan Kaufmann. 193–214.

[Lebowitz, 1986b] Lebowitz, M. 1986b. Integrated learning: Controlling explanation. *Cognitive Science* 10:219–240.

[Lebowitz, 1987] Lebowitz, M. 1987. Experiments with incremental concept formation: UNIMEM. *Machine Learning* 2:103–138.

[Levesque and Brachman, 1987] Levesque, H. J. and Brachman, R. J. 1987. Expressiveness and tractability in knowledge representation and reasoning. *Computational Intelligence* 3:78–92.

[Levinson, 1985] Levinson, R. A. 1985. *A Self-Organizing Retrieval System for Graphs.* Ph.D. Dissertation, University of Texas at Austin.

[Levinson, 1992] Levinson, R. 1992. Pattern associativity and the retrieval of semantic networks. *Computers Math. Applic.* 23(6):573–600.

[Levinson, 1994] Levinson, R. A. 1994. UDS: A universal data structure. Technical Report UCSC-CRL-94-15, University of California, Santa Cruz.

[Lloyd, 1987] Lloyd, J. W. 1987. *Foundations of logic programming.* Springer–Verlag.

[MacGregor and Brill, 1992] MacGregor, R. and Brill, D. 1992. Recognition algorithms for the LOOM classifier. In *Proc. AAAI-92.* The MIT Press. 774–779.

[Markovitch and Scott, 1993] Markovitch, S. and Scott, P. D. 1993. Information filtering: selection mechanisms in learning systems. *Machine Learning* 10:113–151.

[Marr, 1982] Marr, D. 1982. *Vision.* Freeman.

[Matheus et al., 1993] Matheus, C. J.; Chan, P. K.; and Piatetsky-Shapiro, G. 1993. Systems for knowledge discovery in databases. *IEEE Trans. Knowledge and Data Engineering* 5(6):903–913.

[McGregor, 1982] McGregor, J. J. 1982. Backtrack search algorithms and the maximal common subgraph problem. *Software—Practice and Experience* 12:23–34.

[Michalski and Stepp, 1983a] Michalski, R. S. and Stepp, R. E. 1983a. Automated construction of classifications: Conceptual clustering versus numerical taxonomy. *IEEE Trans. Pattern Analysis and Machine Intelligence* 5(4):219–243.

[Michalski and Stepp, 1983b] Michalski, R. S. and Stepp, R. E. 1983b. Learning from observation: conceptual clustering. In Michalski, R.; Carbonell, J.; and Mitchell, T., editors, *Machine Learning.* Tioga. 331–363.

[Mineau and Godin, 1994] Mineau, G. W. and Godin, R. 1994. Automatic structuring of knowledge bases by conceptual clustering. *IEEE Trans. Knowledge and Data Engineering.* To appear.

[Muggleton et al., 1992] Muggleton, S.; Srinivasan, A.; and Bain, M. 1992. Compression, significance and accuracy. In Sleeman, D. and Edwards, P., editors, *Machine Learning: Proceedings of the Ninth International Conference (ML92).* Morgan Kaufmann. 338–347.

[Muggleton and Feng, 1992] Muggleton, S. and Feng, C. 1992. Efficient induction of logic programs. In Muggleton, S., editor, *Inductive Logic Programming.* Academic Press. 281–298.

[Muggleton, 1992] Muggleton, S. 1992. Inductive logic programming. In Muggleton, S., editor, *Inductive Logic Programming.* Academic Press. 3–27.

[Muggleton, 1993] Muggleton, S. 1993. Inductive logic programming: derivations, successes and shortcomings. In Bradzil, P. B., editor, *ECML-93 : Lecture Notes in Artificial Intelligence 667.* Springer Verlag. 21–38.

[Nagy et al., 1988] Nagy, M.; Kozics, S.; Veszpremi, T.; and Bruck, P. 1988. Substructure search on very large files using tree-structured databases. In Warr, W. A., editor, *Chemical Structures: The International Language of Chemistry.* Springer–Verlag. 127–130.

[Nebel, 1990] Nebel, B. 1990. *Reasoning and Revision in Hybrid Representation Systems.* Springer–Verlag.

[Nebel, 1991] Nebel, B. 1991. Terminological cycles: Semantics and computational properties. In Sowa, J. F., editor, *Principles of Semantic Networks.* Morgan–Kaufmann. 331–361.

[Nordhausen and Langley, 1990] Nordhausen, B. and Langley, P. 1990. An integrated approach to empirical discovery. In Shrager, J. and Langley, P., editors, *Computational models of scientific discovery and theory formation.* Morgan Kaufmann. chapter 4, 97–128.

[Okada and Wipke, 1989] Okada, T. and Wipke, W. T. 1989. CLUSMOL: A system for the conceptual clustering of molecules. *Tetrahedron Computer Methodology* 2(4):249–264.

[Okada, 1993] Okada, T. 1993. Similarity and analogy based on discrimination net. In Warr, W., editor, *Chemical Structures 2.* Springer-Verlag. 389–398.

[Parsons, 1989] Parsons, T. J. 1989. Conceptual clustering in relational structures: An application in the domain of vision. In *EWSL89, Proceedings of the Fourth European Working Session on Learning.* 163–177.

[Pentland, 1986] Pentland, A. 1986. Perceptual organization and the representation of natural form. *Artificial Intelligence* 28:293–331.

[Pitt and Reinke, 1988] Pitt, L. and Reinke, R. E. 1988. Criteria for polynomial-time (conceptual) clustering. *Machine Learning* 2:371–396.

[Plotkin, 1971] Plotkin, G. D. 1971. A note on inductive generalisation. *Machine Intelligence* 6:101–124.

[Quinlan, 1990] Quinlan, J. R. 1990. Learning logical definitions from relations. *Machine Learning* 5(3):239–266.

[Rouveirol, 1994] Rouveirol, C. 1994. Flattening and saturation: two representation changes for generalization. *Machine Learning* 14(2):219–232.

[Salton and McGill, 1983] Salton, G. and McGill, M. J. 1983. *Introduction to modern information retrieval.* McGraw-Hill.

[Sammut and Banerji, 1986] Sammut, C. and Banerji, R. B. 1986. Learning concepts by asking questions. In Michalski, R.; Carbonell, J.; and Mitchell, T., editors, *Machine Learning: An Artificial Intelligence Approach*, volume II. Morgan Kaufmann. 167–191.

[Segen, 1990] Segen, J. 1990. Graph clustering and model learning by data compression. In *Proceedings of the Seventh International Conference on Machine Learning.* 93–101.

[Sengupta and Boyer, 1993] Sengupta, K. and Boyer, K. 1993. Information theoretic clustering of large structural modelbases. In *Int. Conf. Computer Vision and Pattern Recognition.* 174–179.

[Shapiro and Haralick, 1982] Shapiro, L. G. and Haralick, R. M. 1982. Organization of relational models for scene analysis. *IEEE Trans. Pattern Analysis and Machine Intelligence* 4(6):595–602.

[Shapiro, 1983] Shapiro, E. Y. 1983. *Algorithmic program debugging.* The MIT Press.

[Stepp and Michalski, 1986] Stepp, R. E. and Michalski, R. S. 1986. Conceptual clustering: inventing goal-oriented descriptions of structured objects. In Michalski, R.; Carbonell, J.; and Mitchell, T., editors, *Machine Learning: An Artificial Intelligence Approach*, volume II. Morgan Kaufmann. 471–498.

[Stepp, 1987] Stepp, R. E. 1987. Machine learning from structured objects. In *Proceedings of the Fourth International Workshop on Machine Learning.* Morgan Kaufmann. 353–363.

[Tennent, 1991] Tennent, R. D. 1991. *Semantics of programming languages.* Prentice–Hall.

[Thompson et al., 1991] Thompson, K.; Langley, P.; and Iba, W. 1991. Using background knowledge in concept formation. In *Proceedings of the Eighth International Workshop on Machine Learning.* Morgan Kaufmann.

[Thompson and Langley, 1989] Thompson, K. and Langley, P. 1989. Organization and retrieval of composite concepts. In *Proceedings of the Case–Based Reasoning Workshop.* Morgan Kaufmann. 329–333.

[Thompson and Langley, 1991] Thompson, K. and Langley, P. 1991. Concept formation in structured domains. In Fisher, D. H.; Pazzani, M.; and Langley, P., editors, *Concept Formation: Knowledge and Experience in Unsupervised Learning.* Morgan Kaufmann. 127–161.

[Vosselman, 1992] Vosselman, G. 1992. *Relational matching: Lecture Notes in Computer Science 628.* Springer-Verlag.

[Wasserman and Lebowitz, 1983] Wasserman, K. and Lebowitz, M. 1983. Representing complex physical objects. *Cognition and Brain Theory* 6(3):333–352.

[Wasserman, 1985] Wasserman, K. 1985. *Unifying representation and generalization: Understanding hierarchically structured objects.* Ph.D. Dissertation, Department of Computer Science, Columbia University.

[Wilcox and Levinson, 1986] Wilcox, C. S. and Levinson, R. A. 1986. A self–organized knowledge base for recall, design, and discovery in organic chemistry. In Pierce, T. H. and Hohne, B. A., editors, *Artificial Intelligence Applications in Chemistry.* American Chemical Society. 209–230.

[Winston, 1975] Winston, P. H. 1975. Learning structural descriptions from examples. In Winston, P. H., editor, *The Psychology of Computer Vision.* McGraw-Hill.

[Wogulis and Langley, 1989] Wogulis, J. and Langley, P. 1989. Improving efficiency by learning intermediate concepts. In *Proc. IJCAI–89.* Morgan–Kaufmann. 657–662.

[Woods and Schmolze, 1992] Woods, W. and Schmolze, J. 1992. The KL-ONE family. *Computers Math. Applic.* 23(6).

[Woods, 1991] Woods, W. 1991. Understanding subsumption and taxonomy: A framework for progress. In Sowa, J. F., editor, *Principles of Semantic Networks*. Morgan–Kaufmann. 45–94.

[Yoshida et al., 1993] Yoshida, K.; Motoda, H.; and Indurkya, N. 1993. Unifying learning methods by colored digraphs. In Tecuci, G.; Kedar, S.; and Kodratoff, Y., editors, *Proceedings of the IJCAI-93 Workshop on Machine Learning and Knowledge Acquisition.* 253–269.

[Zytkow, 1993] Zytkow, J. 1993. Cognitive autonomy in machine discovery. *Machine Learning* 12:7–16. Introduction to Special Issue on Machine Discovery.